

Hacking Articles

Raj Chandel's Blog



Menu

[Home](#) » [Penetration Testing](#) » [VULS- An Agentless Vulnerability Scanner](#)

Penetration Testing

VULS- An Agentless Vulnerability Scanner

October 4, 2020 By Raj Chandel

VULS is an open-source agentless vulnerability scanner that is written In GO Language for Linux Systems. For server Administrator having to perform software updates and security vulnerability analysis daily can be a burden. VULS can be useful or helpful to automate Vulnerability Analysis and to Avoid the burden of manually performing of Vulnerability analysis of the software installed on the system. It uses multiple Vulnerability databases, such as Metasploit, Exploit DB, NVD (National Vulnerability Database).

Table of Content

- **Vuls**
 - Key Features
 - Architecture
- **Prerequisites**
- **Installation & Configuration of Dependencies**
- **Installing Dependencies**
- **Installation & Configuration of GO-CVE-Dictionary**
- **Installation & Configuration of Goval-Dictionary**
- **Installation & Configuration of Gost**

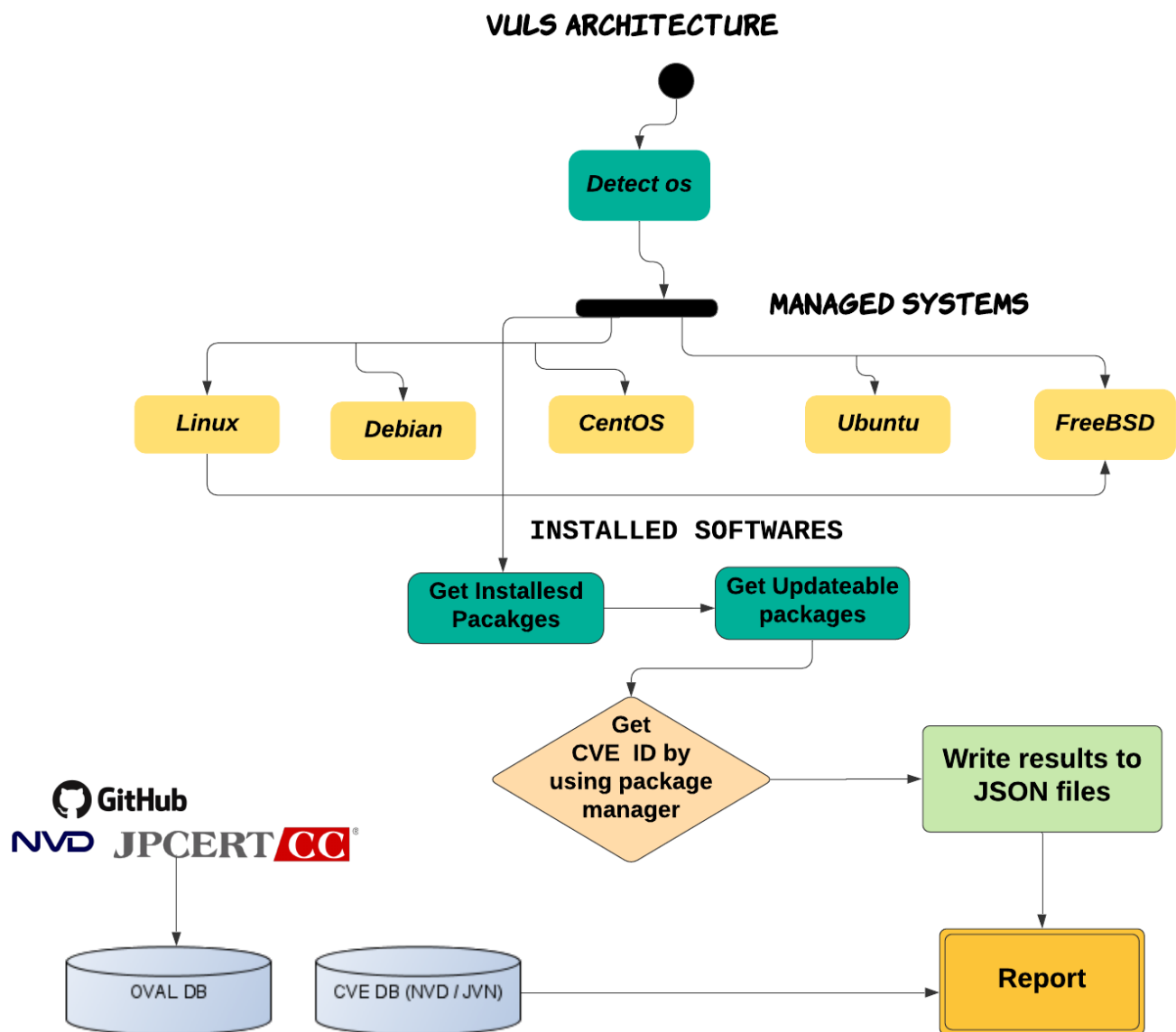
- **Install and Configure VULS**
 - Install and configure VULS repo (GUI) Server
 - Requirements
 - Installation
 - Usage
 - DigestAuth
 - Configuration of TOML file
- **Running Local Scan**
- **Scanning Multiple Remote Host systems**

Vuls Feature & Architecture

Key Features

- VULS provides a way of automating Vulnerability for Linux packages
- VULS can be installed on all Linux based distros for example: – Linux, Ubuntu, Debian, FreeBSD, CentOS, Solaris, and so on...
- VULS has the ability to scan multiple systems at a single time by using SSH protocol and to send reports via Slack or Email.
- VULS uses three Scan modes Fast, Fast Root, and Deep you can select according to Situation or as per your requirements.
- Scan results can be Viewed by using TUI (Terminal user interface) and GUI (Graphical user interface).
- When generating reports VULS prioritizes High Severity Vulnerabilities using an established ranking system from Database.

Architecture



Fast Scan: – It performs scans without root privilege, No dependencies, almost no load on the scan target server.

Fast-root Scan: – It performs scans with root privilege, no dependencies, almost no load on the scan target server.

Deep Scan: – It performs a scan with root privilege, containing all dependencies, almost full load on the scan target server.

Offline Scan Mode: – Fast, Fast-root, and Deep have Offline Scan Mode. VULS can Scan with no internet access with offline scan mode.

Now we see how to install and Configure VULS as a Vulnerability scanner for further investigations.

Let's take a look 🤔!!

Prerequisites

To configure VULS in your Ubuntu platform, there are some prerequisites required for installation.

- Ubuntu 20.04.1 with minimum 4GB RAM and 2 CPU
- SSH Access with Root Privileges
- Firewall Port: – 5111
- Multiple servers running (ubuntu 20.04 or any vulnerable server) if you want to set up VULS to scan remotely.

Installation & Configuration of Dependencies

Let's begin the installation process

Note: – The whole installation process will take a long time to finish so make your self comfortable and begin the installation process.

Installing Dependencies

In this section, we're going to create a folder Vuls-data. VULS uses SQLite to store their vulnerability information so, that we're going to install SQLite, Go programming language, and other dependencies.

We are going to store all VULS related data in the /usr/share/vuls-data directory. To create it run the following as described below.

```
mkdir /usr/share/vuls-data
```

Now we have created the vuls-data folder where we are going to store all data, and this will be our workspace before getting started let's install the required dependencies.

Now, we're going to install

- **SQLite:** – VULS uses SQLite to store its vulnerability information.
- **Debian-goodies:** – it is the check restart utility that provides the information which package needs to be restarted in any moment.
- **GCC:** – GNU compiler collection is a compiler system. GCC is a toolchain and the standard compiler for Unix-like systems.
- **Wget**

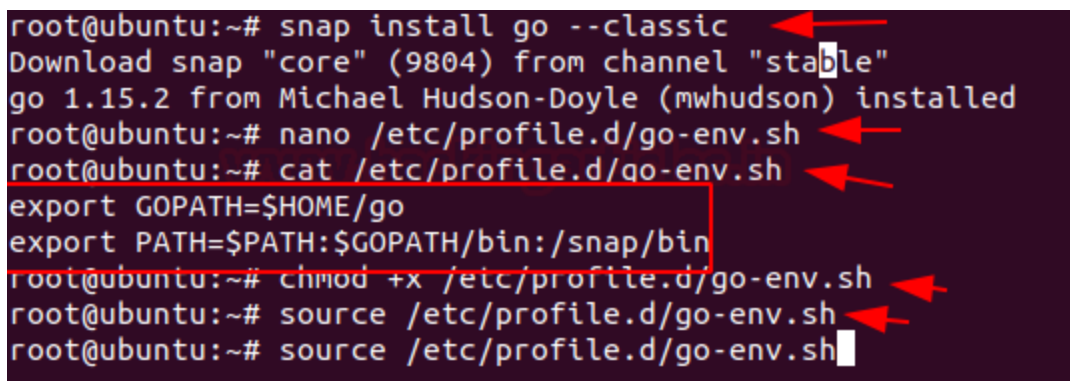

```
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin:/snap/bin
```

Still, the go-env.sh file is not executable. Make it executable by running by changing the permission of that file or directly running by the following command.

```
chmod +x /etc/profile.d/go-env.sh
```

And then reload the environment variables by running the following command.

```
source /etc/profile.d/go-env.sh
```



```
root@ubuntu:~# snap install go --classic
Download snap "core" (9804) from channel "stable"
go 1.15.2 from Michael Hudson-Doyle (mwhudson) installed
root@ubuntu:~# nano /etc/profile.d/go-env.sh
root@ubuntu:~# cat /etc/profile.d/go-env.sh
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin:/snap/bin
root@ubuntu:~# chmod +x /etc/profile.d/go-env.sh
root@ubuntu:~# source /etc/profile.d/go-env.sh
root@ubuntu:~# source /etc/profile.d/go-env.sh
```

Installation & Configuration of Go-CVE-dictionary

Let's download and install go-cve-dictionary. The go-cve-dictionary is a tool that provides access to NVD (National Vulnerability Database). NVD is the US government repository of publicly reported cybersecurity vulnerabilities, that contains vulnerability IDs (CVE — Common Vulnerabilities and Exposures), summaries, and impact analysis, and is available in a machine-readable format. You can access the NVD using the Go package. Then you'll need to run and fetch vulnerability data for VULS to use.

Let's install Go-cve-dictionary under \$GOPATH/src/ by cloning GO packages from GitHub which is made by Kotakanbe and compiling it afterwards.

Let's start it by creating a directory where to store Go-cve-dictionary by running the following command.

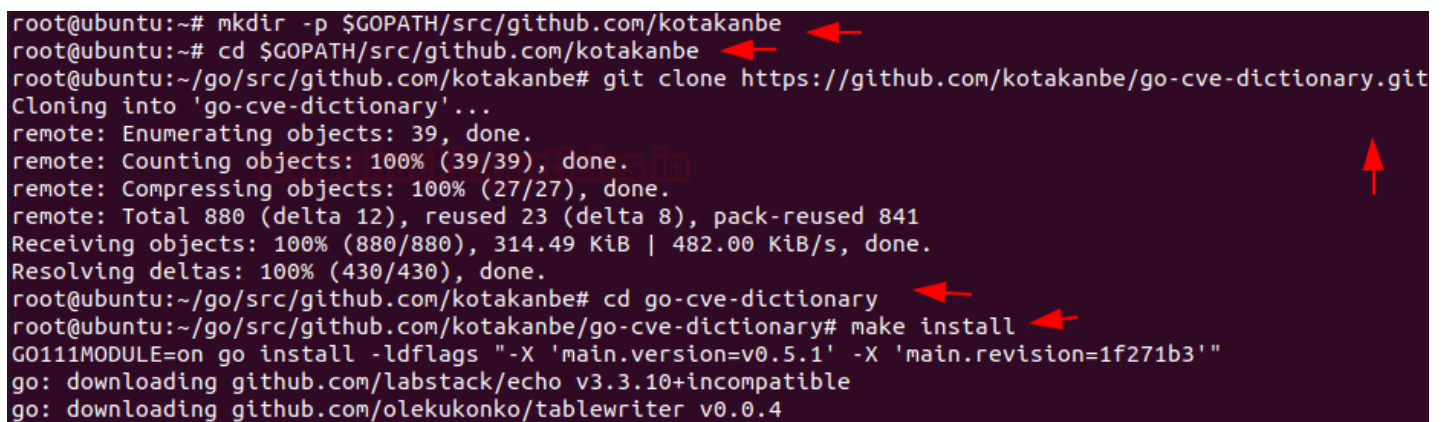
```
mkdir -p $GOPATH/src/github.com/Kotakanbe
```

Navigate to it and clone go-cve-dictionary from GitHub by issuing the following command

```
cd $GOPATH/src/github.com/kotakanbe
git clone https://github.com/kotakanbe/go-cve-dictionary.git
```

And then navigate to the cloned package further then start the installation.

```
cd go-cve-dictionary
make install
```



```
root@ubuntu:~# mkdir -p $GOPATH/src/github.com/kotakanbe
root@ubuntu:~# cd $GOPATH/src/github.com/kotakanbe
root@ubuntu:~/go/src/github.com/kotakanbe# git clone https://github.com/kotakanbe/go-cve-dictionary.git
Cloning into 'go-cve-dictionary'...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 880 (delta 12), reused 23 (delta 8), pack-reused 841
Receiving objects: 100% (880/880), 314.49 KiB | 482.00 KiB/s, done.
Resolving deltas: 100% (430/430), done.
root@ubuntu:~/go/src/github.com/kotakanbe# cd go-cve-dictionary
root@ubuntu:~/go/src/github.com/kotakanbe/go-cve-dictionary# make install
GO111MODULE=on go install -ldflags "-X 'main.version=v0.5.1' -X 'main.revision=1f271b3'"
go: downloading github.com/labstack/echo v3.3.10+incompatible
go: downloading github.com/olekukonko/tablewriter v0.0.4
```

Further then, make it available wide into your system copy it to /usr/local/bin by running below command.

```
cp $GOPATH/bin/go-cve-dictionary /usr/local/bin
```

go-cve dictionary requires a log output directory and logs are generally created under /var/logs/.

Let's create a directory for go-cve-dictionary and the log directory is readable by everyone restrict to the current user by issuing following command.

```
mkdir /var/log/vuls
chmod 700 /var/log/vuls
```

```

root@ubuntu:~/go/src/github.com/kotakanbe/go-cve-dictionary# sudo cp $GOPATH/bin/g
o-cve-dictionary /usr/local/bin
root@ubuntu:~/go/src/github.com/kotakanbe/go-cve-dictionary# mkdir /var/log/vuls
root@ubuntu:~/go/src/github.com/kotakanbe/go-cve-dictionary# chmod 700 /var/log/vu
ls
root@ubuntu:~/go/src/github.com/kotakanbe/go-cve-dictionary#

```

Now fetch vulnerability data from NVD and store it to VULS workspace under /usr/share/vuls-data:

```
for i in `seq 2014 $(date +%Y)`; do sudo go-cve-dictionary fetchnvd -dbpath
```

In my case I'm cloning the cve database from the year 2014 this will download the NVD data from 2014 to the current year you can clone or download data as desired year as you want.

Note: – This command will take a long time to finish, till then go get served you with a Coffee ☕.

```

root@ubuntu:~/go/src/github.com/kotakanbe/go-cve-dictionary# for i in `seq 2014 $(
date +%Y)`; do sudo go-cve-dictionary fetchnvd -dbpath /usr/share/vuls-data/cve.
sqlite3 -years $i; done
INFO[09-11|12:00:49] Fetching... https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.
1-2014.meta
INFO[09-11|12:00:49] Fetching... https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.
1-modified.meta
INFO[09-11|12:00:49] Fetching... https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.
1-recent.meta
INFO[09-11|12:00:51] Fetched... https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1

```

Installation & Configuration of goval-dictionary

Let's download and install "goval-dictionary". The goval-dictionary is a tool that will copy the OVAL (Open Vulnerability and Assessment Language) which is an open language used to express checks for determining whether software vulnerabilities exist on a given system and provides access to the OVAL database for Ubuntu.

The **goval-dictionary** is also written by Kotakanbe so, install goval-dictionary in the same folder that previously you created under "\$GOPATH/src/github.com/Kotakanbe" and then clone the Package from the GitHub by running the following command.

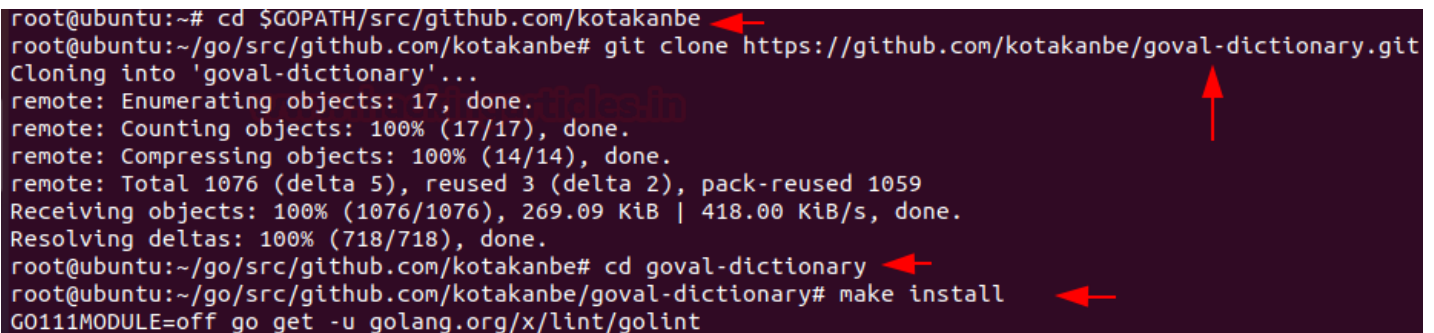
```

cd $GOPATH/src/github.com/kotakanbe
git clone https://github.com/kotakanbe/goval-dictionary.git

```


And then navigate to the cloned package further then compile or install it with “make” by running the following command.

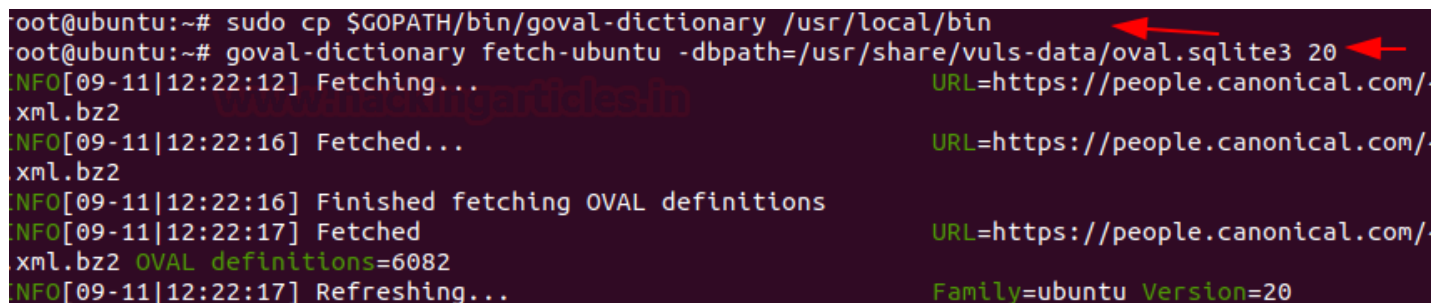
```
cd goval-dictionary
make install
```



```
root@ubuntu:~# cd $GOPATH/src/github.com/kotakanbe
root@ubuntu:~/go/src/github.com/kotakanbe# git clone https://github.com/kotakanbe/goval-dictionary.git
Cloning into 'goval-dictionary'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 1076 (delta 5), reused 3 (delta 2), pack-reused 1059
Receiving objects: 100% (1076/1076), 269.09 KiB | 418.00 KiB/s, done.
Resolving deltas: 100% (718/718), done.
root@ubuntu:~/go/src/github.com/kotakanbe# cd goval-dictionary
root@ubuntu:~/go/src/github.com/kotakanbe/goval-dictionary# make install
GO111MODULE=off go get -u golang.org/x/lint/golint
```

Copy it to /usr/local/bin to make it globally accessible and then after that Fetch the OVAL data for Ubuntu 20.x or another version as per your requirement by running the following command.

```
cp $GOPATH/bin/goval-dictionary /usr/local/bin
goval-dictionary fetch-ubuntu -dbpath=/usr/share/vuls-data/oval.sqlite3 20
```



```
root@ubuntu:~# sudo cp $GOPATH/bin/goval-dictionary /usr/local/bin
root@ubuntu:~# goval-dictionary fetch-ubuntu -dbpath=/usr/share/vuls-data/oval.sqlite3 20
[INFO][09-11|12:22:12] Fetching... URL=https://people.canonical.com/
xml.bz2
[INFO][09-11|12:22:16] Fetched... URL=https://people.canonical.com/
xml.bz2
[INFO][09-11|12:22:16] Finished fetching OVAL definitions
[INFO][09-11|12:22:17] Fetched URL=https://people.canonical.com/
xml.bz2 OVAL definitions=6082
[INFO][09-11|12:22:17] Refreshing... Family=ubuntu Version=20
```

Installation & Configuration of gost

Let's download and install “gost”. Gost is a Debian security Bug tracker that collects all information about the vulnerability status of packages distributed with Debian and provides access to the Debian security bug tracker database.

Let's install this package into a new folder by running the following command:

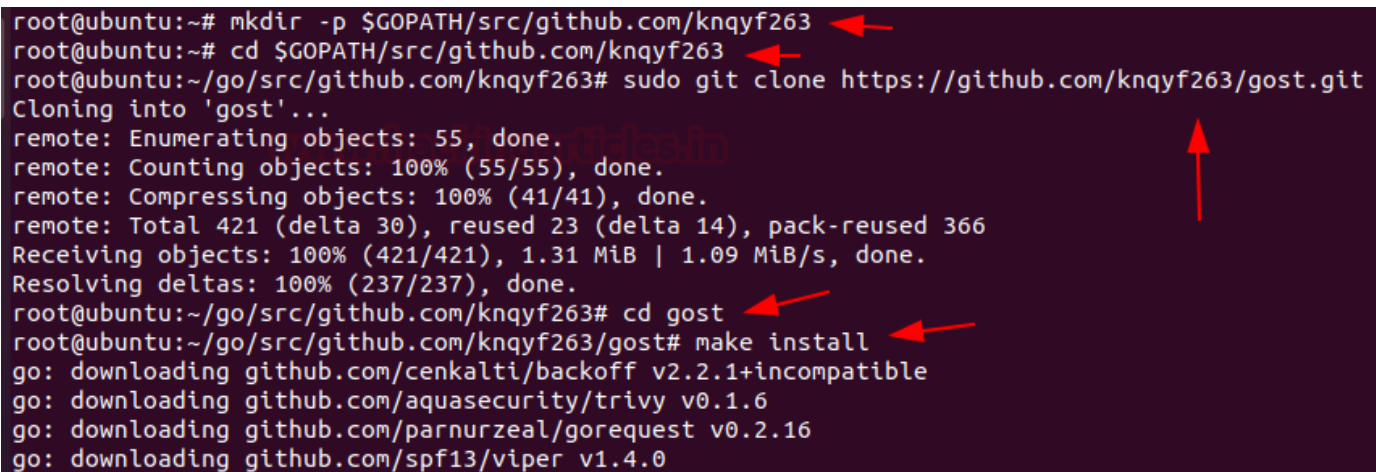
```
mkdir -p $GOPATH/src/github.com/knqyf263
```

Navigate to the folder just you have created after that, clone the gost packages from GitHub, and then make install by running the following command:

```
cd $GOPATH/src/github.com/knqyf263
sudo git clone https://github.com/knqyf263/gost.git
```

After it finishes entering to the cloned package than **“make install”**

```
cd gost
make install
```



```
root@ubuntu:~# mkdir -p $GOPATH/src/github.com/knqyf263
root@ubuntu:~# cd $GOPATH/src/github.com/knqyf263
root@ubuntu:~/go/src/github.com/knqyf263# sudo git clone https://github.com/knqyf263/gost.git
Cloning into 'gost'...
remote: Enumerating objects: 55, done.
remote: Counting objects: 100% (55/55), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 421 (delta 30), reused 23 (delta 14), pack-reused 366
Receiving objects: 100% (421/421), 1.31 MiB | 1.09 MiB/s, done.
Resolving deltas: 100% (237/237), done.
root@ubuntu:~/go/src/github.com/knqyf263# cd gost
root@ubuntu:~/go/src/github.com/knqyf263/gost# make install
go: downloading github.com/cenkalti/backoff v2.2.1+incompatible
go: downloading github.com/aquasecurity/trivy v0.1.6
go: downloading github.com/parnurzeal/gorequest v0.2.16
go: downloading github.com/spf13/viper v1.4.0
```

Don't forget to make it accessible globally and then link its database to the /usr/share/vuls-data by running the following command:

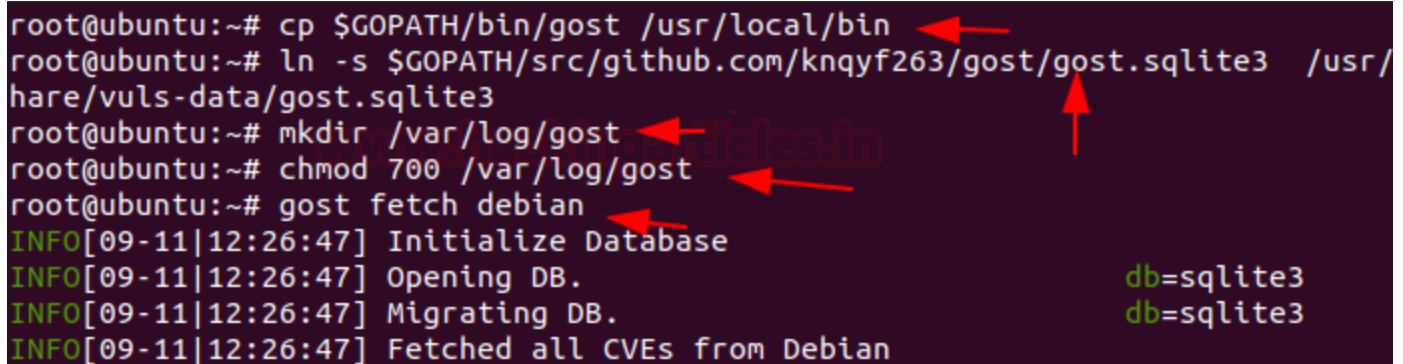
```
cp $GOPATH/bin/gost /usr/local/bin
ln -s $GOPATH/src/github.com/knqyf263/gost/gost.sqlite3 /usr/share/vuls-data/
```

Create a log file directory for gost it requires access to the log output directory and then restricts access to the current user by using the following command:

```
mkdir /var/log/gost
chmod 700 /var/log/gost
```

And then, fetch the Debian security tracker data by issuing the following command:

```
gost fetch debian
```



```
root@ubuntu:~# cp $GOPATH/bin/gost /usr/local/bin
root@ubuntu:~# ln -s $GOPATH/src/github.com/knqyf263/gost/gost.sqlite3 /usr/
share/vuls-data/gost.sqlite3
root@ubuntu:~# mkdir /var/log/gost
root@ubuntu:~# chmod 700 /var/log/gost
root@ubuntu:~# gost fetch debian
INFO[09-11|12:26:47] Initialize Database
INFO[09-11|12:26:47] Opening DB.
INFO[09-11|12:26:47] Migrating DB.
INFO[09-11|12:26:47] Fetched all CVEs from Debian
```

Install & Configure VULS

We have installed all required Dependencies of VULS. Now you can download and install Vuls from source code. Afterwards, you'll configure the VULS reps server which is the GUI interface of the VULS.

Let's Create a new directory that contains the path to the Vuls repository, by issuing the following command:

```
mkdir -p $GOPATH/src/github.com/future-architect
```

Navigate to the created directory then Clone Vuls from GitHub by running the following command:

```
cd $GOPATH/src/github.com/future-architect
git clone https://github.com/future-architect/vuls.git
```

Enter to the Package Folder and then compile and install by running the following command:

```
cd vuls
make install
```

```
root@ubuntu:~# mkdir -p $GOPATH/src/github.com/future-architect
root@ubuntu:~#
root@ubuntu:~# cd $GOPATH/src/github.com/future-architect
root@ubuntu:~/go/src/github.com/future-architect# git clone https://github.com/future-architect/vuls.git
Cloning into 'vuls'...
remote: Enumerating objects: 63, done.
remote: Counting objects: 100% (63/63), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 6145 (delta 20), reused 16 (delta 3), pack-reused 6082
Receiving objects: 100% (6145/6145), 5.94 MiB | 2.09 MiB/s, done.
Resolving deltas: 100% (4265/4265), done.
root@ubuntu:~/go/src/github.com/future-architect# cd vuls
root@ubuntu:~/go/src/github.com/future-architect/vuls# make install
go: downloading github.com/knqyf263/go-deb-version v0.0.0-20190517075300-09fca494f03d
go: downloading github.com/emersion/go-smtp v0.13.0
go: downloading github.com/aws/aws-sdk-go v1.33.21
```

Also, don't forget to make it accessible globally

```
cp $GOPATH/bin/vuls /usr/local/bin
```

Hmm 😊 !! you have successfully installed VULS in your system

Install & Configure VULS repo server (GUI)

VulsRepo is an awesome OSS Web UI for Vuls. With VulsRepo you can analyze the scan results like an Excel pivot table.

Requirements

To configure VULS in your Ubuntu platform, there are some prerequisites required for installation.

- [future-architect/Vuls](#) >= v0.4.0
- Web Browser: Google Chrome or Firefox

Installation

In manner to install Vuls-repo server in your Ubuntu platform follow the steps as stated below

Step1. Installation

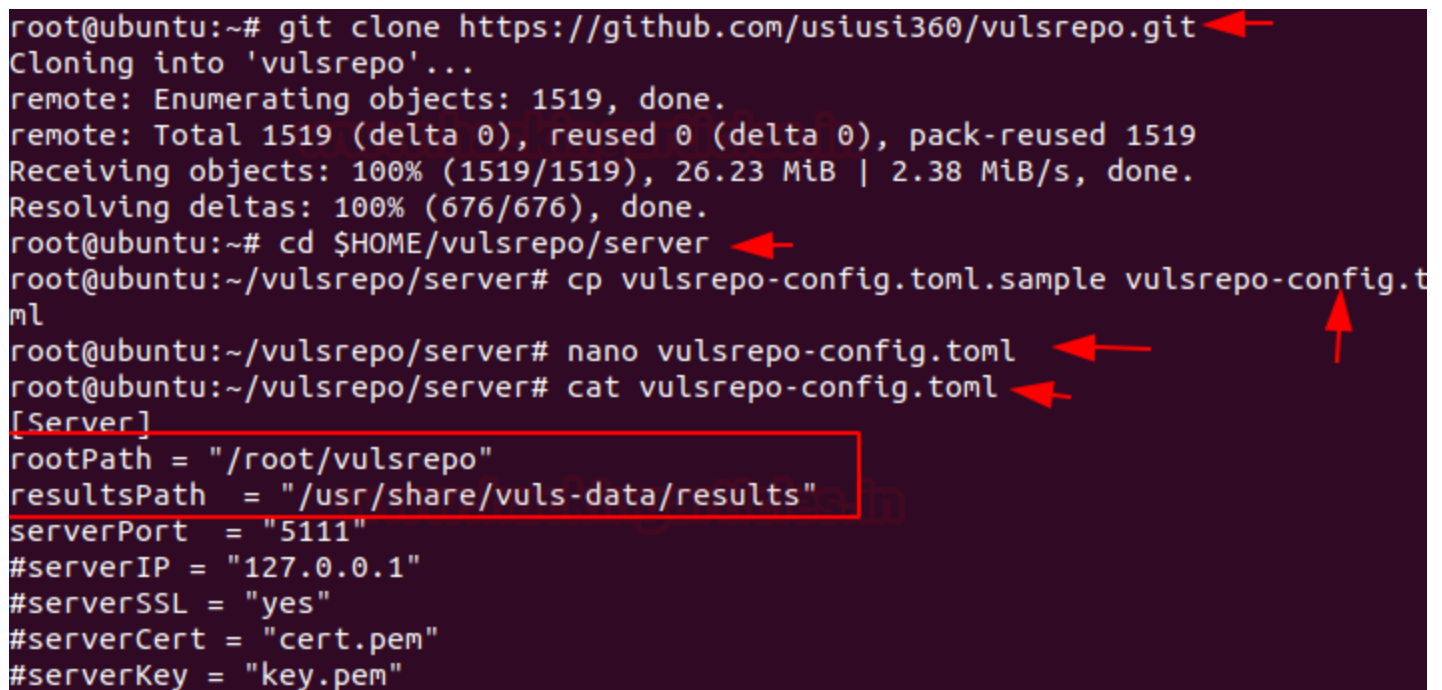
Clone the vuls-repo from GitHub by running the following command:

```
cd $HOME
git clone https://github.com/usiusi360/vulsrepo.git
```

Step 2. Change the setting of vulsrepo-server

Set Path according to your environment.

```
cd $HOME/vulsrepo/server
cp vulsrepo-config.toml.sample vulsrepo-config.toml
nano vulsrepo-config.toml
[Server]
rootPath = "/root/vulsrepo"
resultsPath = "/usr/share/vuls-data/results"
serverPort = "5111"
```



```
root@ubuntu:~# git clone https://github.com/usiusi360/vulsrepo.git
Cloning into 'vulsrepo'...
remote: Enumerating objects: 1519, done.
remote: Total 1519 (delta 0), reused 0 (delta 0), pack-reused 1519
Receiving objects: 100% (1519/1519), 26.23 MiB | 2.38 MiB/s, done.
Resolving deltas: 100% (676/676), done.
root@ubuntu:~# cd $HOME/vulsrepo/server
root@ubuntu:~/vulsrepo/server# cp vulsrepo-config.toml.sample vulsrepo-config.toml
root@ubuntu:~/vulsrepo/server# nano vulsrepo-config.toml
root@ubuntu:~/vulsrepo/server# cat vulsrepo-config.toml
[Server]
rootPath = "/root/vulsrepo"
resultsPath = "/usr/share/vuls-data/results"
serverPort = "5111"
#serverIP = "127.0.0.1"
#serverSSL = "yes"
#serverCert = "cert.pem"
#serverKey = "key.pem"
```

Step 3. Start vulsrepo-server

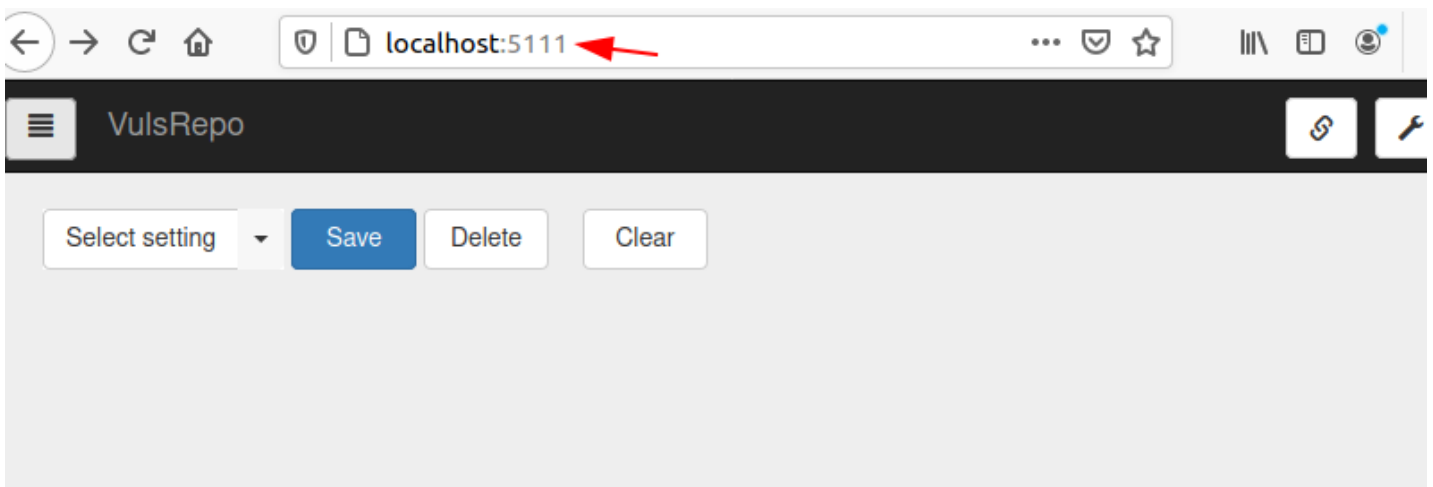
Start the vulsrepo-server by executing the below command under the directory

```
HOME/vulsrepo/server
cd $HOME/vulsrepo/server
./vulsrepo-server
```

```
root@ubuntu:~/vulsrepo/server# ./vulsrepo-server  
2020/09/12 11:49:45 main.go:153: INFO: RootPath Load: /root/vulsrepo  
2020/09/12 11:49:45 main.go:160: INFO: ResultsPath Load: /usr/share/vuls-data  
results  
2020/09/12 11:49:45 main.go:128: Start: Listening port: :5111  
2020/09/12 11:50:06 main.go:202: /
```

You can also verify whether it is running or not by opening the below URL, you need to make sure port 5111 is open on your server firewall and then you can access vulsrepo-server on the web interface at

`http://localhost:5111`



Nice 😊 !! As you can see it is successfully installed

Step 4. Always activate vulsrepo-server

In Case: SystemV (/etc/init.d)

Copy the startup file. Change the variable according to the environment.

```
cp $HOME/vulsrepo/server/scripts/vulsrepo.init /etc/init.d/vulsrepo  
chmod 755 /etc/init.d/vulsrepo  
nano /etc/init.d/vulsrepo
```

```
root@ubuntu:~# cp $HOME/vulsrepo/server/scripts/vulsrepo.init /etc/init.d/vulsrepo  
root@ubuntu:~# chmod 755 /etc/init.d/vulsrepo  
root@ubuntu:~# nano /etc/init.d/vulsrepo  
root@ubuntu:~#
```

And then make changes conf file as per your environment

```

GNU nano 4.8 /etc/init.d/vulsrepo
#!/bin/bash
#chkconfig: 2345 85 15
#description: VulsRepo
#OS RedHat6 / CentOS6 / AmazonLinux

# change according to the environment
USER="root"
BIN_DIR="/root/vulsrepo/server"

# source function library
. /etc/rc.d/init.d/functions

RETVAL=0

start() {
    echo -n "Starting VulsRepo: "
    su ${USER} -c "/usr/bin/nohup ${BIN_DIR}/vulsrepo-server >/dev/null 2>&"
    RETVAL=$?
    if [ $RETVAL == 0 ]; then
        echo success
    fi
}

```

In Case of systemd (systemctl)

Copy the startup file. Change the variables according to the environment.

```

sudo cp $HOME/vulsrepo/server/scripts/vulsrepo.service /lib/systemd/system/vulsrepo.service
nano /lib/systemd/system/vulsrepo.service

```

```

root@ubuntu:~# cp $HOME/vulsrepo/server/scripts/vulsrepo.service /lib/systemd/system/vulsrepo.service
root@ubuntu:~# nano /lib/systemd/system/vulsrepo.service

```

And then make a change in the conf file as per your environment as shown below

```

GNU nano 4.8 /lib/systemd/system/vulsrepo.service Modified
[Unit]
Description=vulsrepo daemon
Documentation=https://github.com/usiusi360/vulsrepo

[Service]
ExecStart = /root/vulsrepo/server/vulsrepo-server
ExecRestart = /bin/kill -WINCH ${MAINPID} ; /root/vulsrepo/server/vulsrepo-server
ExecStop = /bin/kill -WINCH ${MAINPID}
Restart = no
Type = simple
User = root
Group = vuls-group

[Install]
WantedBy = multi-user.target

```



```
start vulsrepo-server  
systemctl start vulsrepo
```

Usage

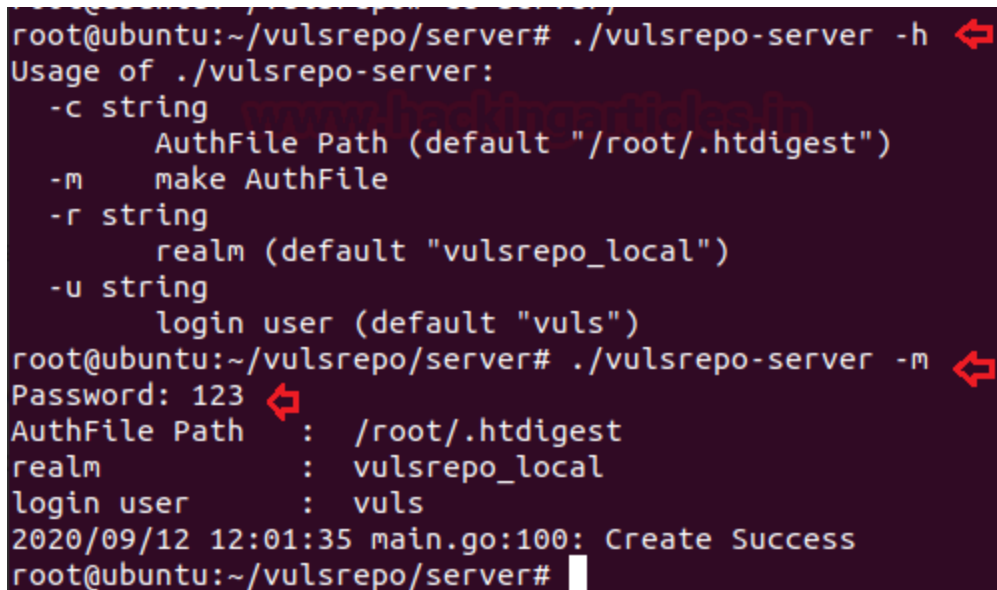
Access the browser

```
http://<server-address>:5111
```

DigestAuth

create an authentication file to perform digest authentication,

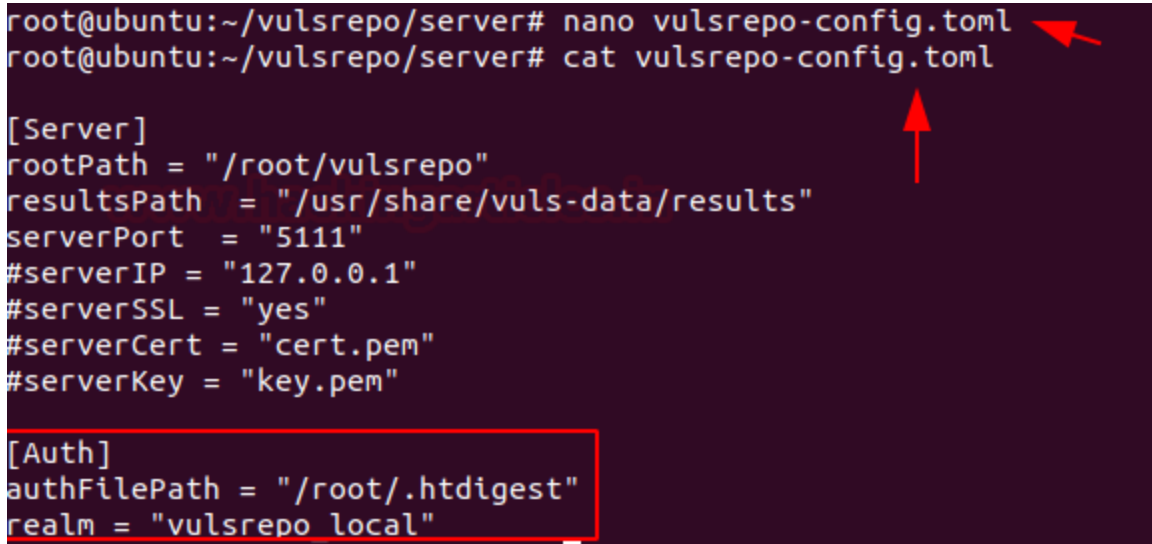
```
./vulsrepo-server -h  
./vulsrepo-server -m
```



```
root@ubuntu:~/vulsrepo/server# ./vulsrepo-server -h  
Usage of ./vulsrepo-server:  
-c string  
    AuthFile Path (default "/root/.htdigest")  
-m      make AuthFile  
-r string  
    realm (default "vulsrepo_local")  
-u string  
    login user (default "vuls")  
root@ubuntu:~/vulsrepo/server# ./vulsrepo-server -m  
Password: 123  
AuthFile Path   : /root/.htdigest  
realm           : vulsrepo_local  
login user      : vuls  
2020/09/12 12:01:35 main.go:100: Create Success  
root@ubuntu:~/vulsrepo/server#
```

Edit vulsrepo-config.toml

```
nano vulsrepo-config.toml
```

```
root@ubuntu:~/vulsrepo/server# nano vulsrepo-config.toml
root@ubuntu:~/vulsrepo/server# cat vulsrepo-config.toml

[Server]
rootPath = "/root/vulsrepo"
resultsPath = "/usr/share/vuls-data/results"
serverPort = "5111"
#serverIP = "127.0.0.1"
#serverSSL = "yes"
#serverCert = "cert.pem"
#serverKey = "key.pem"

[Auth]
authFilePath = "/root/.htdigest"
realm = "vulsrepo local"
```

Use SSL

Create a self-signed certificate by issuing the following command

```
openssl genrsa -out key.pem 2048
openssl req -new -x509 -sha256 -key key.pem -out cert.pem -days 3650
```

Edit **vulsrepo-config.toml** file as shown below by running the following command

```
nano vulsrepo-config.toml
```

```

root@ubuntu:~# openssl genrsa -out key.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
root@ubuntu:~# openssl req -new -x509 -sha256 -key key.pem -out cert.pem -days 3650
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Delhi
Locality Name (eg, city) []:delhi
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ignite
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:Ignite
Email Address []:info@ignitetechnologies.in
root@ubuntu:~# nano vulsrepo-config.toml
root@ubuntu:~# cd vulsrepo/server/
root@ubuntu:~/vulsrepo/server# nano vulsrepo-config.toml
root@ubuntu:~/vulsrepo/server# cat vulsrepo-config.toml

[Server]
rootPath = "/root/vulsrepo"
resultsPath = "/usr/share/vuls-data/results"
serverPort = "5111"
#serverIP = "127.0.0.1"
serverSSL = "yes"
serverCert = "cert.pem"
serverKey = "key.pem"

```

Start vulsrepo-server

Restart Vulsrepo-server by running the following command:

```
systemctl restart vulsrepo-server
```

And then visit the web interface, enter the login credentials that you created during the installation process to access the GUI interface. Once you logged in then you will have your VULS GUI Dashboard ready to set fire on the Vulnerability 😊.

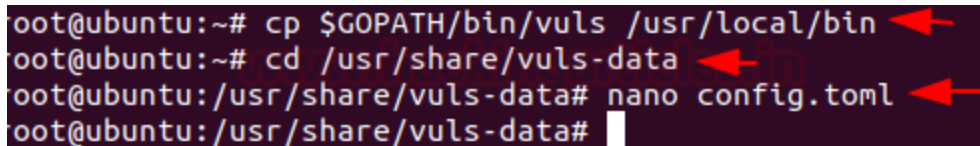
Configuration of TOML file

Now, it's time to create a configuration file for Vuls. Navigate back to /usr/share/vuls-data:

```
cd /usr/share/vuls-data
```

Vuls stores its configuration in a TOML file, which is config.toml. Create by issuing the following command:

```
nano config.toml
```

A terminal window with a dark background and light-colored text. It shows a series of commands being executed in a root shell on an Ubuntu system. The commands are: 'cp \$GOPATH/bin/vuls /usr/local/bin', 'cd /usr/share/vuls-data', and 'nano config.toml'. Red arrows point to the end of each command line. The prompt changes from '~#' to '/usr/share/vuls-data#' after the directory change.

```
root@ubuntu:~# cp $GOPATH/bin/vuls /usr/local/bin
root@ubuntu:~# cd /usr/share/vuls-data
root@ubuntu:/usr/share/vuls-data# nano config.toml
root@ubuntu:/usr/share/vuls-data#
```

And then Enter the following configuration:

```
[cveDict]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/cve.sqlite3"
[ovalDict]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/oval.sqlite3"
[gost]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/gost.sqlite3"
[servers]
[servers.localhost]
host = "localhost"
port = "local"
scanMode = [ "fast" ]
#scanMode = ["fast", "fast-root", "deep", "offline"]
```

Then save and close the file.

```
GNU nano 4.8 config.toml
SQLite3Path = "/usr/share/vuls-data/gost.sqlite3"

[servers]

[servers.localhost]
host = "localhost"
port = "local"
scanMode = [ "fast" ]
#scanMode = ["fast", "fast-root", "deep", "offline"]
```

Ok 😊 !! you have successfully created a conf.toml file

To test the validity of the configuration file, run the following command:

```
vuls configtest
```

Congratulations!! You've installed and configured Vuls to scan the local server on your Ubuntu Platform 😊.

Running local Scan

Exited? let's do it 😊 !!

The default scan mode, if not explicitly specified, is fast.

To run a scan, execute the following command:

```
vuls scan
```

```

root@ubuntu:/usr/share/vuls-data# vuls scan
[Sep 12 12:12:19] INFO [localhost] Start scanning
[Sep 12 12:12:19] INFO [localhost] config: /usr/share/vuls-data
[Sep 12 12:12:19] INFO [localhost] Validating config...
[Sep 12 12:12:19] INFO [localhost] Detecting Server/Container
[Sep 12 12:12:19] INFO [localhost] Detecting OS of servers...
[Sep 12 12:12:19] INFO [localhost] (1/1) Detected: localhost: u
[Sep 12 12:12:19] INFO [localhost] Detecting OS of containers..
[Sep 12 12:12:19] INFO [localhost] Checking Scan Modes...
[Sep 12 12:12:19] INFO [localhost] Detecting Platforms...
[Sep 12 12:12:20] INFO [localhost] (1/1) localhost is running o
[Sep 12 12:12:20] INFO [localhost] Detecting IPS identifiers...
[Sep 12 12:12:20] INFO [localhost] (1/1) localhost has 0 IPS in
[Sep 12 12:12:20] INFO [localhost] Scanning vulnerabilities...
[Sep 12 12:12:20] INFO [localhost] Scanning vulnerable OS packa
[Sep 12 12:12:20] INFO [localhost] Scanning in fast mode

One Line Summary
=====
[Reboot Required] localhost ubuntu20.04 1645 installed

To view the detail, vuls tui is useful.
To send a report, run vuls report -h.
root@ubuntu:/usr/share/vuls-data# vuls tui
[Sep 12 12:12:42] INFO [localhost] Validating config...

```

Wow !! As we can see it scanned the whole system and generated a report

Wait this is not enough... Let's what's inside the report

To check the report on TUI (Terminal based user interface) issue the following command

vuls tui

Vuls divides the generated report view into four panels as stated below:

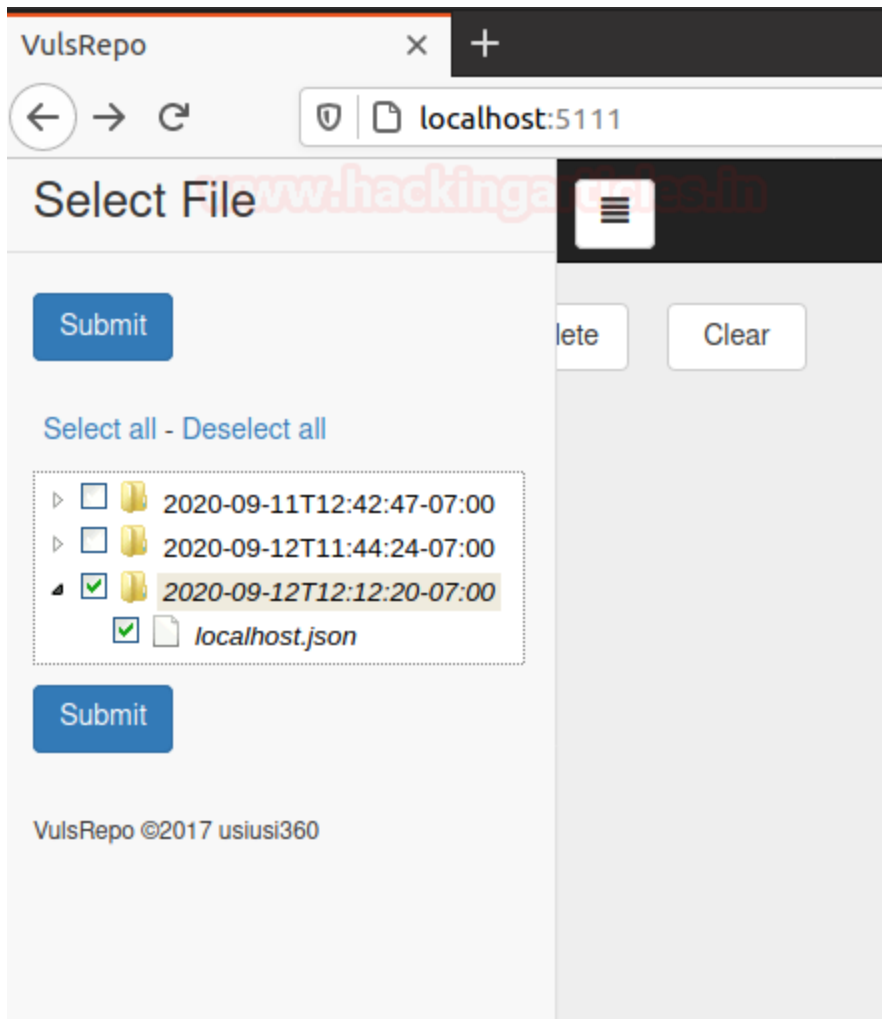
- Scanned hosts: located on the upper left, lists hosts that Vuls scanned.
- Found vulnerabilities: located right of the host's list, shows the vulnerabilities that VULS found in installed packages.
- Vulnerabilities information: it is up of the left part of the screen, that shows detailed information about the vulnerability, pulled from the databases.
- Vulnerable packages: the located right of the detailed information, shows the affected packages and their versions.

localhost (ubuntu20.04)	1] CVE-2016-1585 9.8 AV:N 2] CVE-2017-8283 9.8 AV:N 3] CVE-2018-20839 9.8 AV:N 4] CVE-2019-18604 9.8 AV:N 5] CVE-2020-24977 9.8 AV:N 6] CVE-2019-19814 9.3 AV:N 7] CVE-2020-15656 9.3 AV:N
CVE-2016-1585 ===== CVSS Scores ----- 9.8/CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H CRITICAL nvd 7.5/AV:N/AC:L/Au:N/C:P/I:P/A:P HIGH nvd 6.9/- MEDIUM ubuntu Summary ----- In all versions of AppArmor mount rules are accidentally widened when compiled. (ubuntu) Mitigation ----- - (unknown) Links ----- * https://nvd.nist.gov/vuln/detail/CVE-2016-1585	Affected Packages, Processes ===== * apparmor-2.13.3-7ubuntu5.1 -> Not fixed yet

Aha 🤔 !! It's hilarious

Let's check the How the GUI shows these results

Get back to the GUI Dashboard and then mark and submit the generated report that you want to view as shown below



And then see the magic 🪄 !!

CVSS Severity ▾ CVSS Score ▾																											
			CVSS Severity	Low							Medium										High						
			CVSS Score		1.9	2.1	2.6	3.6	4.3	4.4	4.6	4.7	4.9	5	5.1	5.8	6	6.2	6.4	6.5	6.8	6.9	7.1	7.2	7.5	7.8	9.0
CveID	CweID	Summary	CVSS Score Type																								
CVE-2009-5080	None	Unknown	Unknown																								
CVE-2012-1093	None	Unknown	Unknown																								
CVE-2012-1096	None	Unknown	Unknown																								
CVE-2012-2663	None	Unknown	Unknown																								
CVE-2012-6655	None	Unknown	Unknown																								
CVE-2013-4235	None	Unknown	Unknown																								
CVE-2013-7445	None	Unknown	Unknown																								
CVE-2014-3495	CWE-295	duplcity 0.6.24 has improper verification of SSL certificates	nvd											1													
CVE-2015-3416	CWE-119	The sqlite3VXPrintf function in printf.c in SQLite before 3.8.9 does not properly handle precision and width values during floating-point conversions, which allows context-dependent attackers to cause a denial of service (integer overflow and stack-based buffer overflow) or possibly have unspecified other impact via large integers in a crafted printf function call in a SELECT statement.	nvd																						1		
CVE-2015-5237	CWE-119	protobuf allows remote authenticated attackers to cause a heap-based buffer	nvd																	2							

As we can see it converted to JSON report into GUI with detailed information. By tapping on CVE ids you can more information about their vulnerability.

Also, you can filter this report as per your known by dragging the required report from the Heatmap section to Count as shown below.

www.hackingarticles.in

www.h

Count	ServerName	Container
Packages		ServerName
CVSS Severity		Container
CveID		
Changelog		
Summary		
Packages	CVSS Severity	CveID
Changelog	Summary	
accounts-service	Unknown	CVE-2012-6655
Changelog	None	Unknown
apparmor	High	CVE-2016-1585
Changelog	None	In all versions of AppArmor mount rules are accidentally widened when compiled.
bash	High	CVE-2019-18276
Changelog	None	An issue was discovered in disable_priv_mode in shell.c in GNU Bash through 5.0 patch 11. By default, if Bash is run with its effective UID not equal to its real UID, it will drop privileges by setting its effective UID to its real UID. However, it does so incorrectly. On Linux and other systems that support "saved UID" functionality, the saved UID is not dropped. An attacker with command execution in the shell can use "enable -f" for runtime loading of a new builtin, which can be a shared object that calls setuid() and therefore regains privileges. However, binaries running with an effective UID of 0 are unaffected.

Let's make it more informative by applying filters as shown below:

CveID	CVSS Severity	Packages	DetectionMethod	CVSS Score Type	PackageVer	NewPackageVer	NotFixedYet
CVE-2009-5080	Unknown	groff-base	OvalMatch	Unknown	1.22.4-4build1-	-	true
CVE-2012-1093	Unknown	xorg	OvalMatch	Unknown	1:7.7+19ubuntu14-	-	true
CVE-2012-1096	Unknown	network-manager	OvalMatch	Unknown	1.22.10-1ubuntu2.1-	-	true
CVE-2012-2663	Unknown	iptables	OvalMatch	Unknown	1.8.4-3ubuntu2-	-	true
CVE-2012-6655	Unknown	accountsservice	OvalMatch	Unknown	0.6.55-0ubuntu12~20.04.1-	-	true
CVE-2013-4235	Unknown	login	OvalMatch	Unknown	1:4.8.1-1ubuntu5.20.04-	-	true
		passwd	OvalMatch	Unknown	1:4.8.1-1ubuntu5.20.04-	-	true
CVE-2013-7445	Unknown	linux-image-5.4.0-42-generic	OvalMatch	Unknown	5.4.0-42.46-	-	true
CVE-2014-3495	Medium	duplcity	OvalMatch	nvd	0.8.11.1612-1-	-	true
CVE-2015-3416	High	sqlite	OvalMatch	nvd	2.8.17-15fakesync1build1-	-	true
CVE-2015-5237	Medium	libprotobuf17	OvalMatch	nvd	3.6.1.3-2ubuntu5-	-	true
		python3-protobuf	OvalMatch	nvd	3.6.1.3-2ubuntu5-	-	true
CVE-2015-8553	Low	linux-image-5.4.0-42-generic	OvalMatch	nvd	5.4.0-42.46-	-	true
CVE-2015-9019	Medium	libxslt1.1	OvalMatch	nvd	1.1.34-4-	-	true
CVE-2016-1585	High	apparmor	OvalMatch	nvd	2.13.3-7ubuntu5.1-	-	true
CVE-2016-2568	Medium	policykit-1	OvalMatch	nvd	0.105-26ubuntu1-	-	true

Scanning Multiple remote host systems

Step 1: Enable to SSH from localhost

Vuls doesn't support SSH password authentication. So we have to use SSH key-based authentication. Create a key pair on the localhost then copy the id_rsa.pub key to authorized_keys on the remote host.

On Localhost:

```
ssh-keygen -t rsa
```

Copy /ssh/id_rsa.pub key to the clipboard.

And go to the **Remote Host** and issue the following command:

```
mkdir ~/.ssh
chmod 700 ~/.ssh
nano ~/.ssh/authorized_keys
```

and then Paste the rsa.pub key from the clipboard to ~/.ssh/authorized_keys and then follow the below steps:

```
chmod 600 ~/.ssh/authorized_keys
```

Come back to the Localhost:

And also, we need to confirm that the host keys of the remote scan target has been registered in the known_hosts of the localhost. Further, then we need to add the remote host's Host Key to \$HOME/.ssh/known_hosts, log in to the remote host through SSH before scanning.

```
ssh root@192.168.29.219 -i ~/.ssh/id_rsa
```

where 192.168.29.219 is the IP of remote Host

Step 2: Configure (config.toml) as shown below

```
cd /usr/share/vuls-data
nano config.toml
[servers.ignite]
host      = "192.168.29.219"
port      = "22"
user      = "root"
keyPath   = "/root/.ssh/id_rsa"
```

A screenshot of a terminal window with a dark background. At the top, it shows 'GNU nano 4.8' on the left and 'confi' on the right. The main content is the configuration for the '[servers.ignite]' section. The lines are: 'host = "192.168.29.219"', 'port = "22"', 'user = "root"', 'keyPath = "/root/.ssh/id_rsa"', and 'scanMode = ["deep"] # "fast", "fast-root" or "deep"'. There are red arrows pointing to the end of each line. A large, semi-transparent watermark 'hackingarticles.in' is visible across the center of the terminal output.

Check and verify config.toml and settings on the server before scanning:

```
vuls configtest
```

Start scanning remote host by issuing the below command:

```
vuls scan
```

```
root@ubuntu:/usr/share/vuls-data# vuls scan
[Sep 14 12:59:52] INFO [localhost] Start scanning
[Sep 14 12:59:52] INFO [localhost] config: /usr/share/vuls-data/config.toml
[Sep 14 12:59:52] INFO [localhost] Validating config...
[Sep 14 12:59:52] INFO [localhost] Detecting Server/Container OS...
[Sep 14 12:59:52] INFO [localhost] Detecting OS of servers...
[Sep 14 12:59:53] INFO [localhost] (1/1) Detected: ignite: ubuntu 20.04
[Sep 14 12:59:53] INFO [localhost] Detecting OS of containers...
```

Congratulation 😊!! As you can see you have been successfully scanned your remote host. Let's check the generated report on the GUI dashboard.

VulsRepo

Select setting

Save

Delete

Clear

Filter OFF

Heatmap

Count

CVSS Severity

CVSS Score

Family

Release

CveID

Packages

NotFixedYet

CweID

ScanTime

ServerName

Container

ScanTime	ServerName	Container	CVSS Severity	Low			Medium										High						
			CVSS Score	1.9	2.1	2.6	3.6	4.3	4.4	4.6	4.7	4.9	5	5.1	5.8	6	6.2	6.4	6.5	6.8	6.9	7.1	7.2
2020-09-14T12:59:59-07:00	ignite [Reboot Required]	None		2	10	1	1	100	1	3	2	10	28	1	6	1	1	3	2	23	1	2	5
Totals				2	10	1	1	100	1	3	2	10	28	1	6	1	1	3	2	23	1	2	5

By applying more filters, you can make you can hunt or investigate Vulnerable packages deeper.

Let's end Here !! 😊

Author – Vijay is a Certified Ethical Hacker, Technical writer and Penetration Tester at Hacking Articles. Technology and Gadget freak. Contact [Here](#)

◀ PREVIOUS POST

PowerGrid: 1.0.1 Vulnhub Walkthrough

NEXT POST ▶

Firefox for Pentester: Privacy and Protection Add-ons

One thought on “VULS- An Agentless Vulnerability Scanner”



[kota](#)

October 6, 2020 at 7:42 am

Hi, I am an author of Vuls, Thanks for the good explanatory material.

usiusi360/vulsrepo is NOT maintained anymore.

We should use a maintained repository: ishiDASCo/vulsrepo

<https://vuls.io/docs/en/vulsrepo.html>

Comments are closed.

Join Our Training Program





Categories

Cryptography & Steganography

CTF Challenges

Cyber Forensics

Database Hacking

Footprinting

Hacking Tools

Kali Linux

Nmap

Others

Password Cracking

Penetration Testing

Pentest Lab Setup

Privilege Escalation

Red Teaming

Social Engineering Toolkit

Uncategorized

Website Hacking

Window Password Hacking

Wireless Hacking

Wireless Penetration Testing

Archives

Select Month



You may like

Best Alternative of Netcat Listener

64-bit Linux Assembly and Shellcoding

